Jack McCorkindale 71415038 George Hampton 16434986

Application Structure *i*

The application flow is primarily centered on the class Game. This communicates with the classes Island, Player, and Display. When the application is run, it generates all the items, advice and islands which are used by the game. Both items and advice are generated from a text file in the classes Item and Store respectively. They are stored statically as it groups them in one place and makes the code more organised. The islands are stored in an array list rather than statically as the location data can be changed throughout different playthroughs.

The Player class is a child of the Ship class as originally there was an enemy class. This changed when it was revealed that the player had all the attributes that the enemy did which resulted in the enemy being reduced back into ship. The inheritance remained as during an event they both had slightly different functionality. Player stored all the information relating to the current user. This includes their name, ship's name, gold, items, location, luck, number of crew, and capacity. The player also stores the games logbook.

The player's logbook contains an ArrayList of entries which stores all the instances of Entry that is created through the course of the game. Every Entry records the day that every event happens. Each possible event has a builder function that adds the appropriate information to the entry which will be formatted and printed if it is a non-null value.

The ArrayList of islands that are generated at the start are hardcoded in, however, changing this to be procedurally generated would not be that difficult due to how Island is created. Every island has a list of routes to other islands, a store, a name and a location. The location is mostly used in conjunction with other islands to determine the distance between them. The store on the island was implemented incorrectly as it is always created by the event of the player arriving at the island.

Store has stock, advice, price modifiers, and island attributes. In hindsight, the store class could have functioned better as an extension of the island class.

The item class was designed to be abstract with children Cargo and Card. These both have different functions in the context of the game and require different functions. Cargo would have the ability to modify the ship while Card had the ability to modify dice rolls in an event.

Testing

The majority of unit testing occurred on functions which control the logical flow of the game. This did not include most functions that contained 1 or 2 lines such as getters and setters. The coverage for these cases varied between classes with those like Card, with more logic, less randomness, and less text output, having a higher coverage. The overall coverage of the project was around 76% excluding the display functions which were tested visually throughout production, as JUnit tests do not work with GUI components. Components of the graphics were implemented in many of the main classes, lowering the overall coverage. The unit tests that were created attempted to test all types of input that would be received, that is both valid and invalid, with an

emphasis on boundary values. As all input from the player was received through the functions Game.getInt() or getNames() invalid input testing was reduced significantly.

Thoughts and feedback

Both partners agree that overall the project was relatively successful as both have learned a significant amount about how to program in java and use object orientated programing. With the limited knowledge at the beginning there are many things in retrospect that could have been improved. The time management of the project wasn't as good as it could have been as the majority of the code was written in the first week. This resulted in a large period of time where not much was done each week due to apparent near finished state of the application.

Retrospective and improvements

The partners worked well as a team with the ability to distribute tasks amongst each other. There was an element of specialisation that occurred due to the strengths of each person. Jack took a larger role in the development of the command line project while George took the leading role in the graphical application.
One of the problems with the project that was faced was testing of the random variables as the outcome couldn't be determined easily before running the code. To improve this in the next project that involves random elements it would be a good idea to pass the value through as a variable. This would allow more deterministic testing.
When starting the project the GUI experience that both partners had was limited which resulted in inaccuracies in how information was displayed. If the project was done again the usage of the function System.out.println() would be reduced within separate functions instead opting for returning an output string.
The original design of the application was that most classes would run through Game due to the limited experience with object orientated programing. Later when this was changed classes were not properly refactored as they should have been. For example the class Island and the class Store could have been more closely related and perhaps have some aspect of inheritance. As the project is set up Island stores Store and passes itself into the Store constructor. This was due to a change in how the logbook system worked requiring the location that each item was bought/sold be determined as well as getting the price for each item.
As a next step for project management, having a stricter deadline system would create an improvement in the work done each week. As most of the design was completed in the first week, progress faltered through the project resulting in a rush at the end.

Contribution

Both patterns agreed that the distribution of work done by each partner is; Jack with 63% with approximately 105 hours of work contributed to the project and George with 37% with approximately 70 hours of work contributed to the project.

---

[i] See UML Class Diagram.PDF within this zip folder.